Koordination verteilter Software-Komponenten

Roland Wismüller

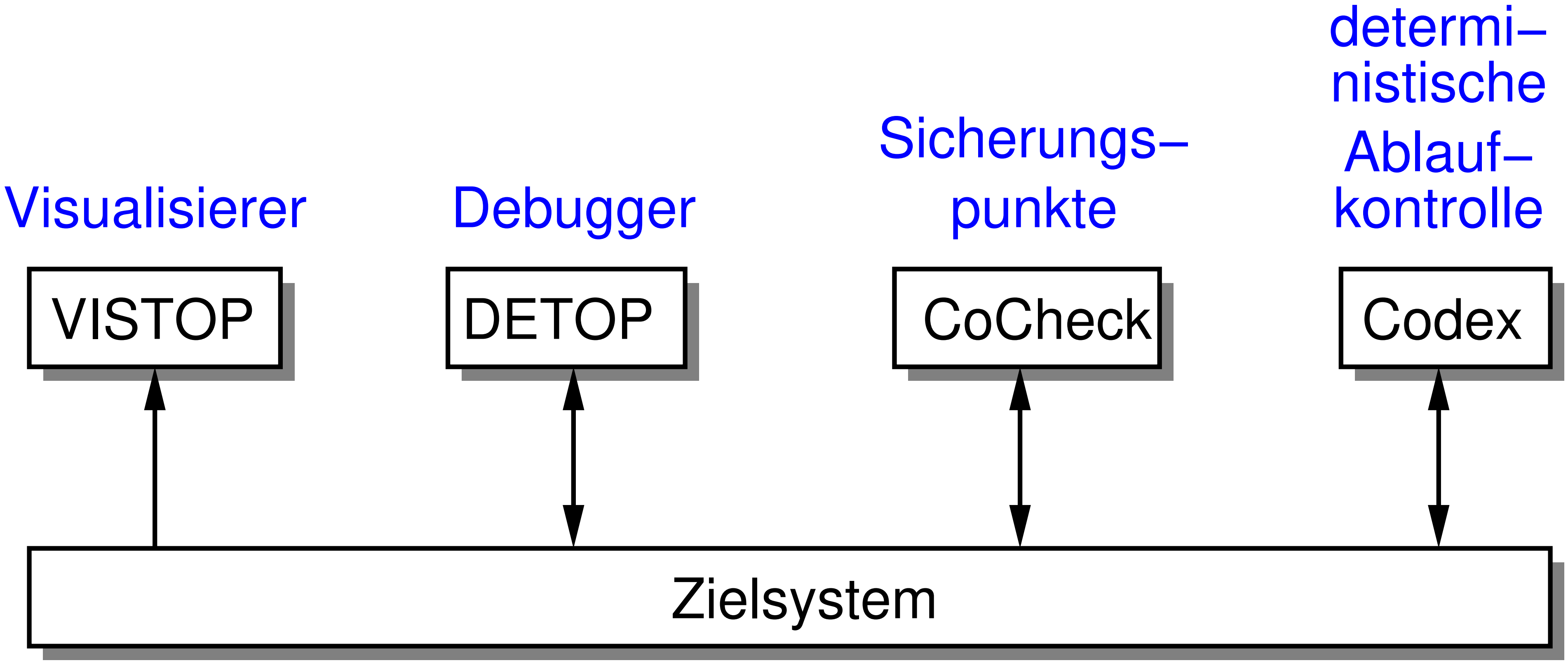
LRR-TUM
Technische Universität München

1. Motivation: Eine Werkzeugumgebung

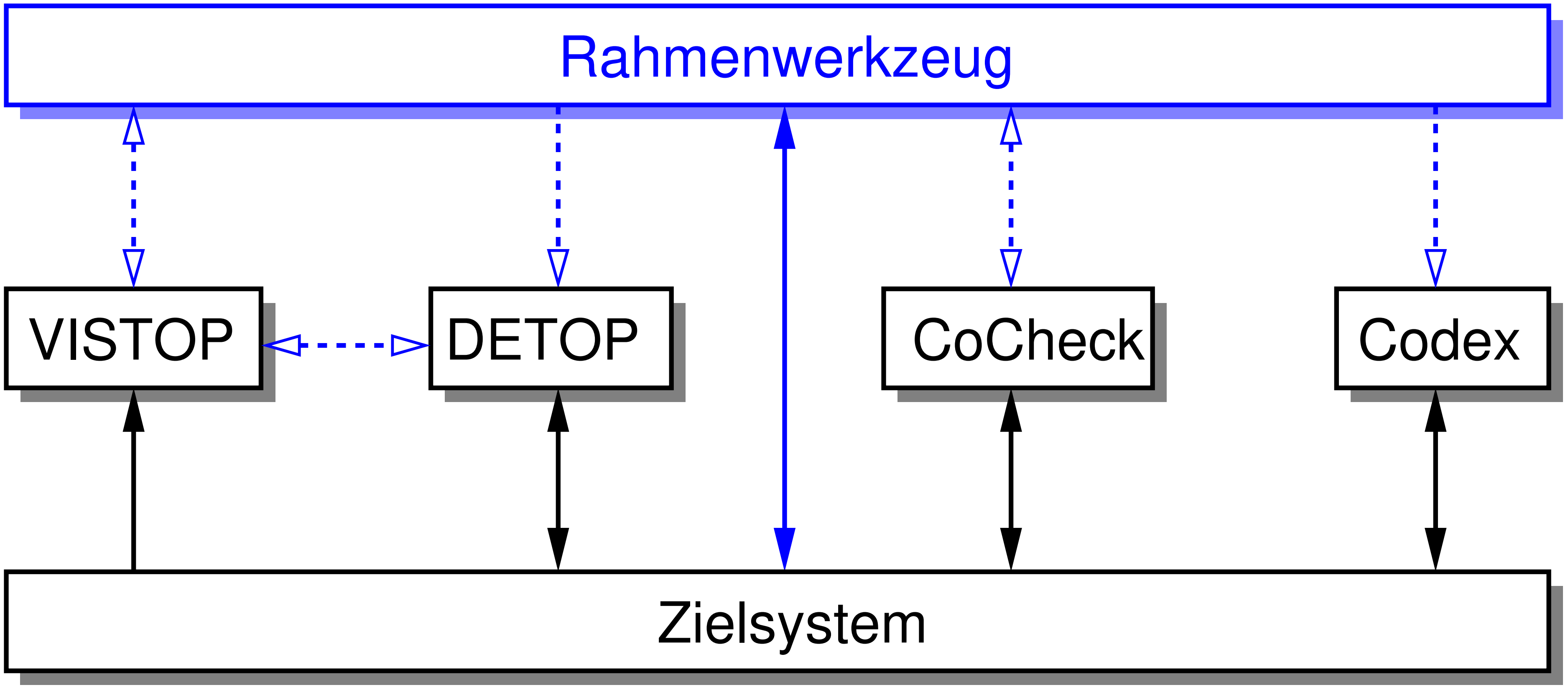
Hintergrund:

- Entwicklung von Online-Werkzeuge am LRR-TUM
  - Quellsprach-Debugger DETOP
  - Ablaufvisualisierer VISTOP
  - Leistungsanalyse, deterministische Ausführung, Sicherungspunkterstellung, ...
- Wunsch: Kombination von Werkzeugen
  - z.B. Visualisierer + Debugger
    - Überblick über Ablaufverhalten
    - detaillierte Untersuchung

Aufbau der Werkzeugumgebung



Aufbau der Werkzeugumgebung



Ziel: Kooperation zwischen den Komponenten

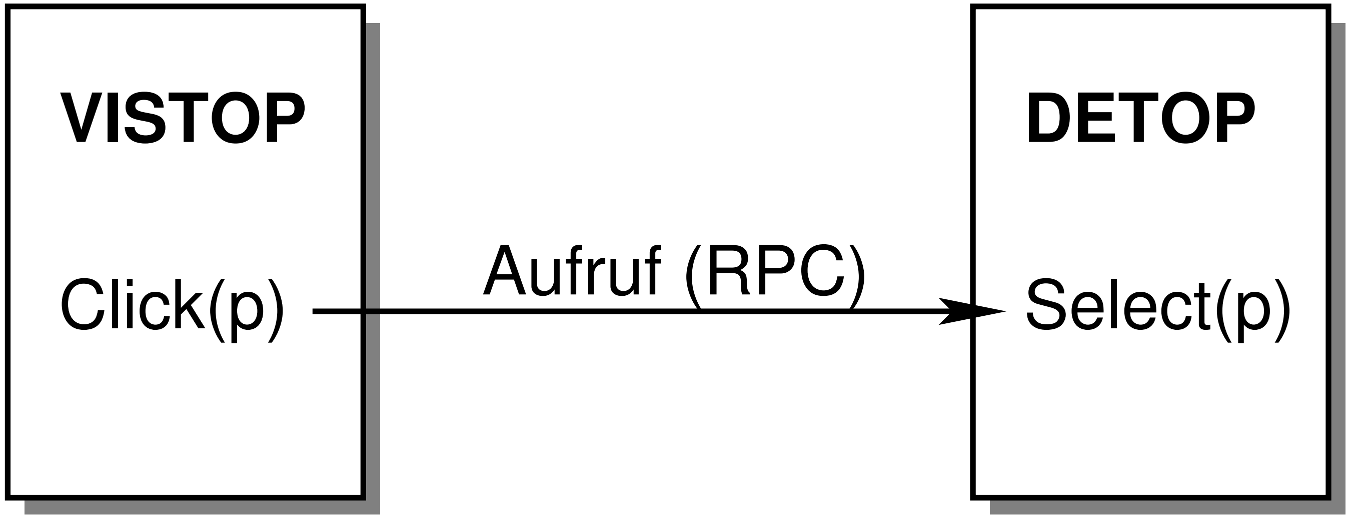
- Z.B. synchronisierte Auswahl, konsistente Darstellung
- Im weiteren Sinne: Steuerung durch Rahmenprogramm

Allgemeine Fragestellung:

- Erstellung von Software-Umgebungen aus vorhandenen Komponenten
  - mit Benutzerinteraktion
  - aktiv, mit eigenständigem Verhalten (im Ggs. zu Servern)
  - unabhängig voneinander entwickelt und nutzbar
  - keine Modifikation der Komponenten

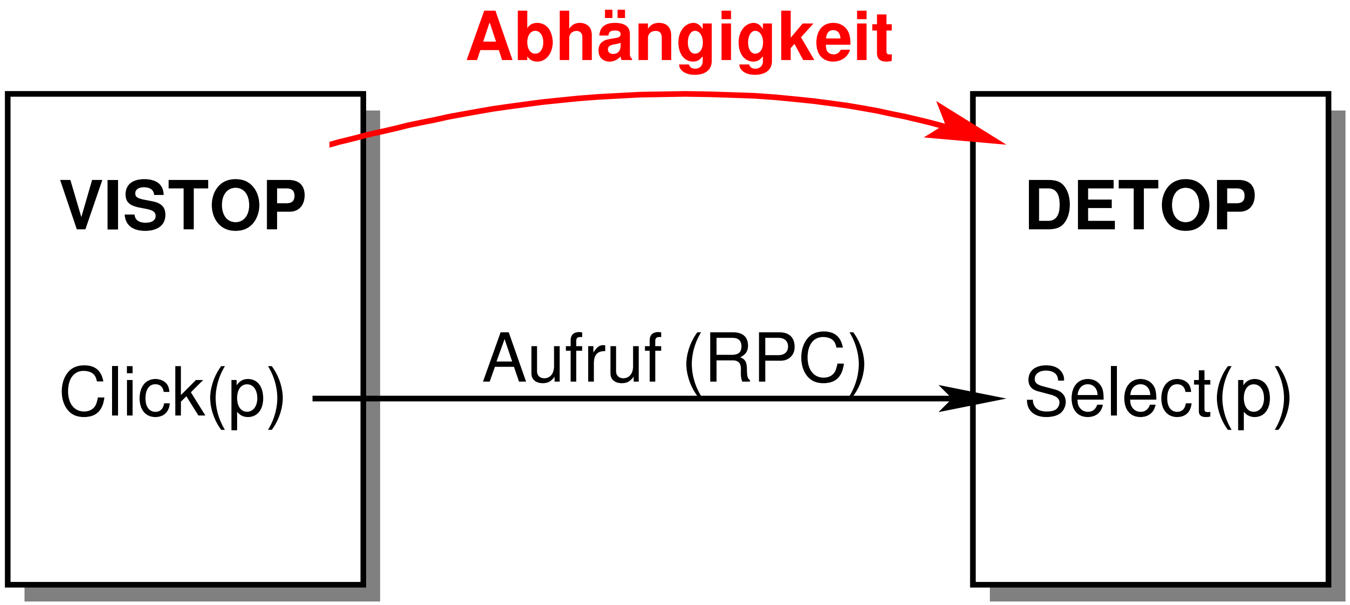
Feste Verdrahtung

- Z.B. RPC, CORBA, PCTE



Feste Verdrahtung

- Z.B. RPC, CORBA, PCTE

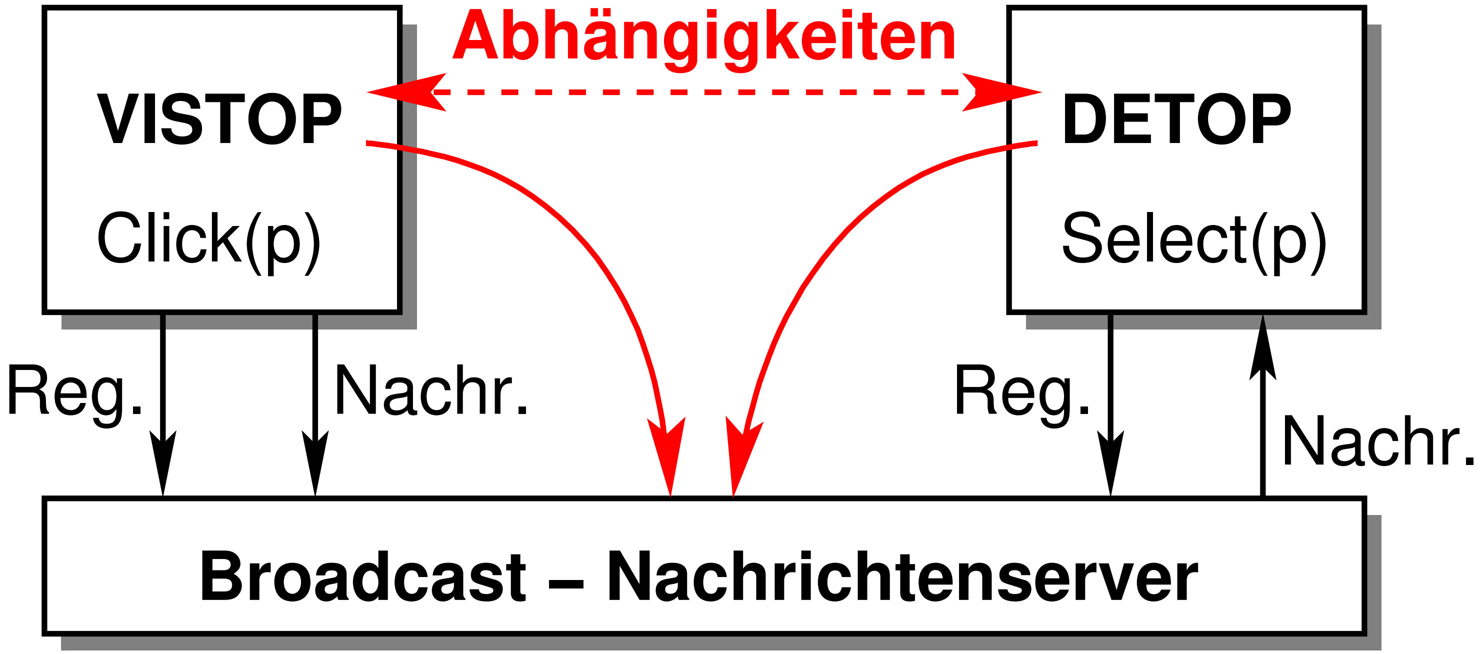


Broadcast-Nachrichtenserver

- Z.B. ToolTalk, SoftBench

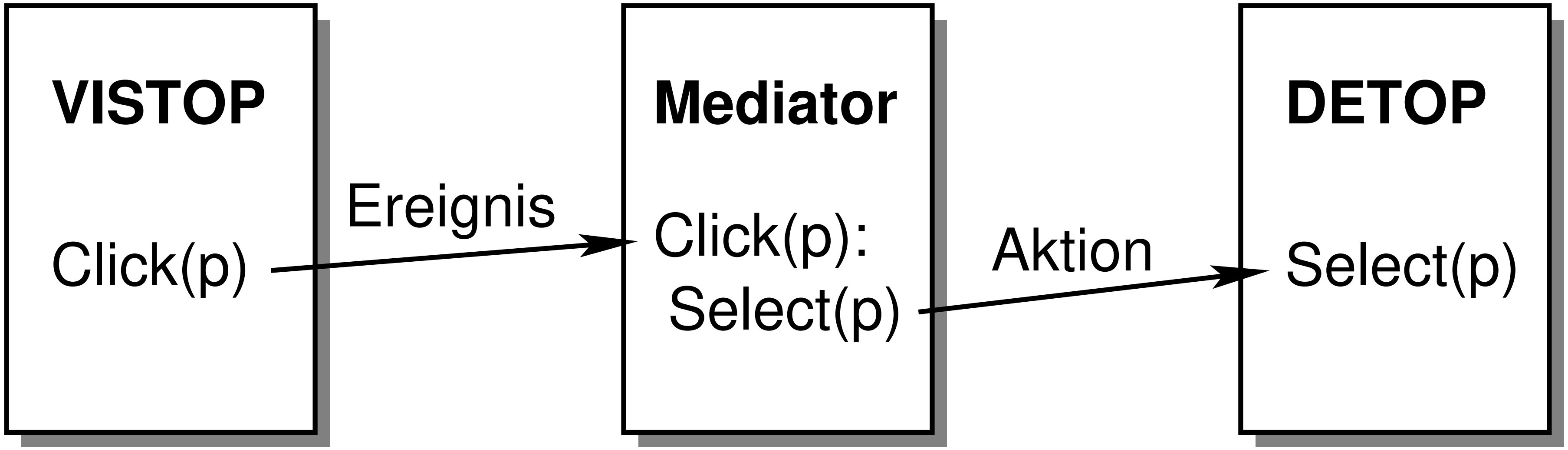
Broadcast-Nachrichtenserver

- Z.B. ToolTalk, SoftBench



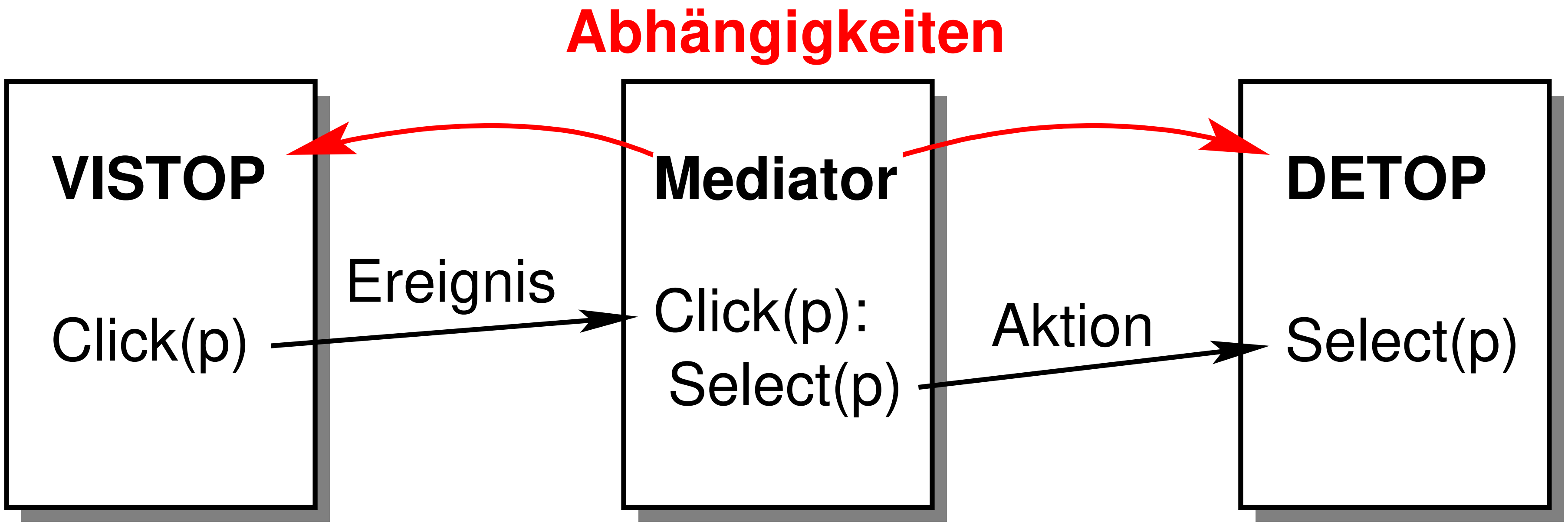
Mediatoren / Interaktoren

- Z.B. ToolBus, Manifold, CHAIMS, TIL



Mediatoren / Interaktoren

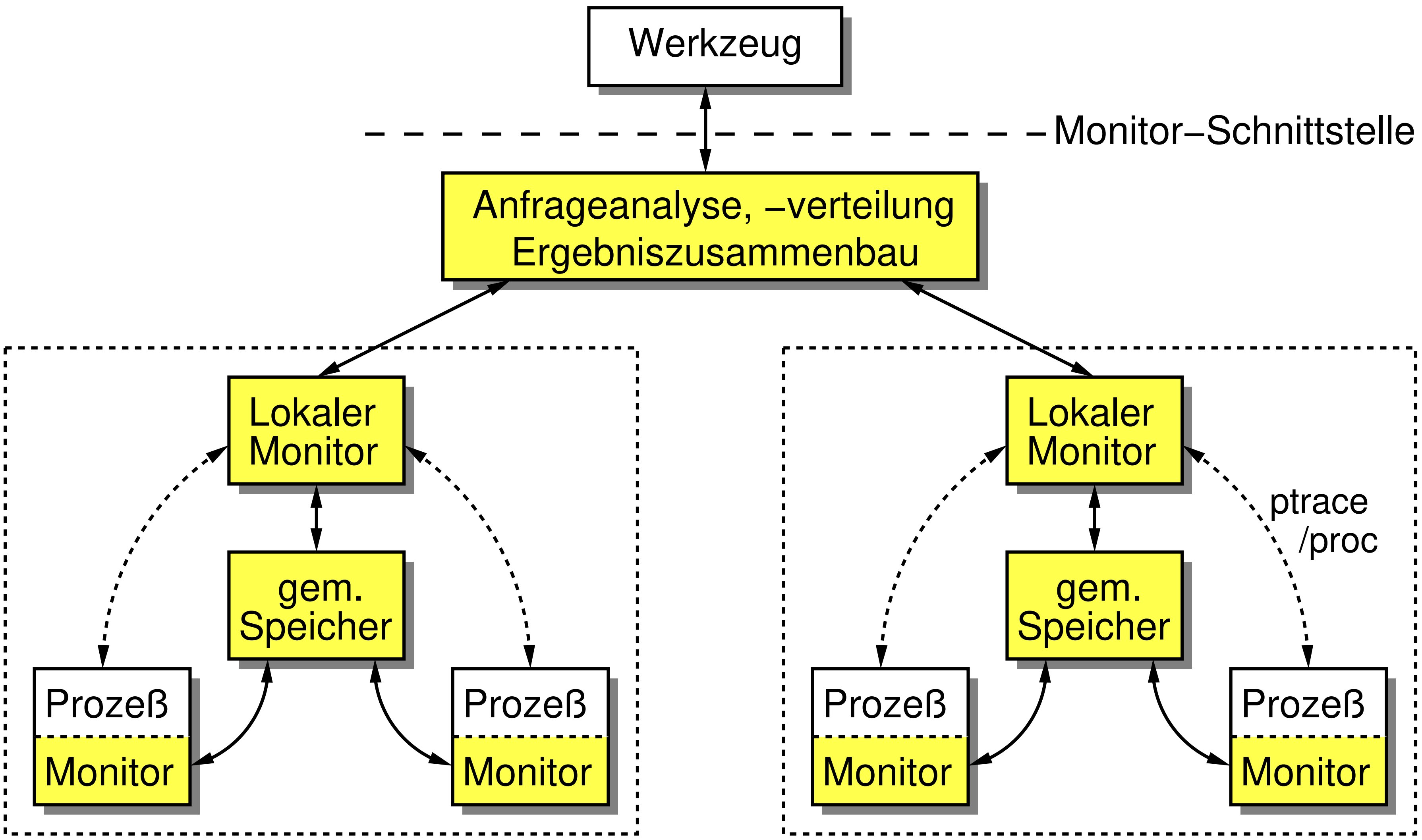
- Z.B. ToolBus, Manifold, CHAIMS, TIL



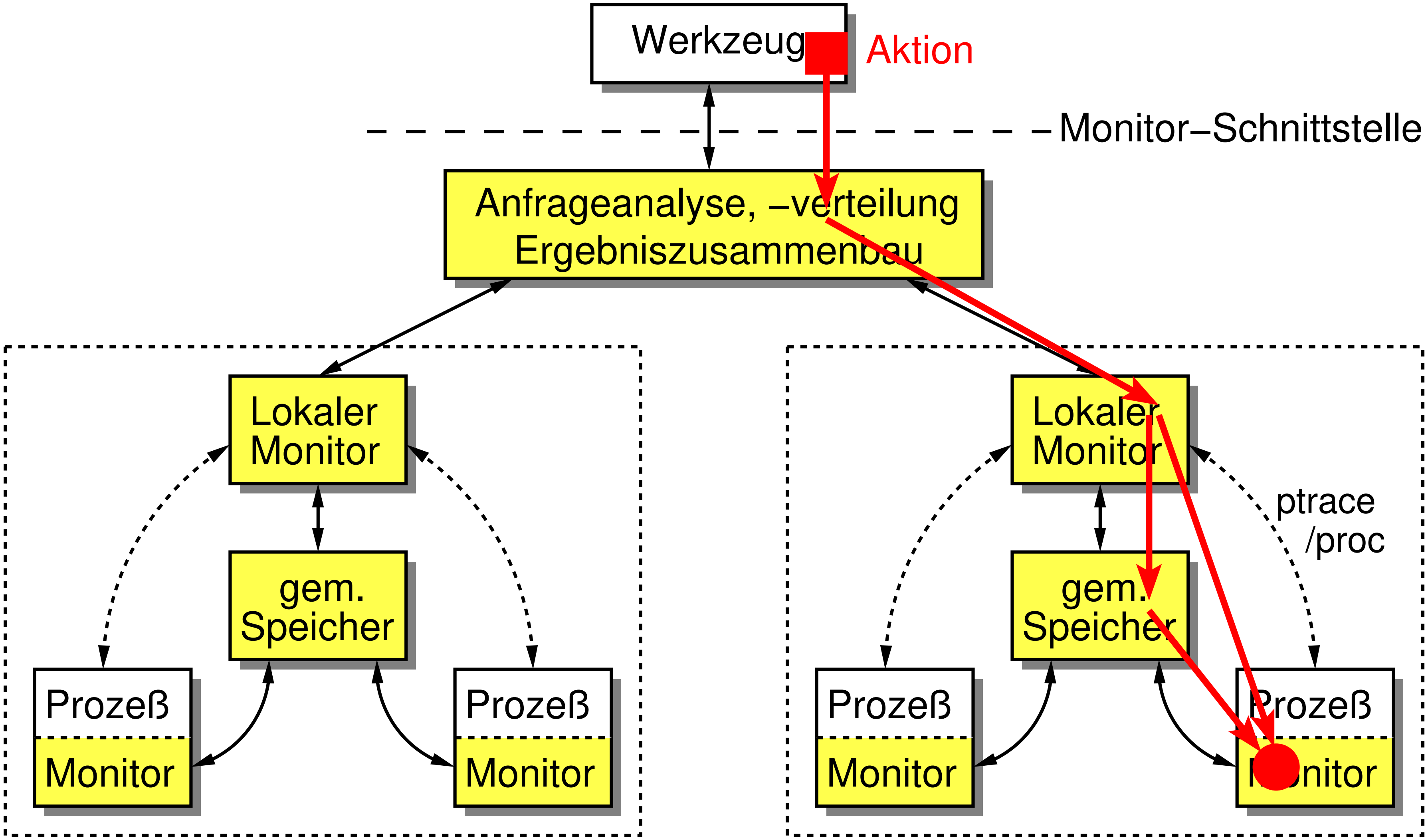
Problem:
Komponenten müssen a-priori Schnittstellen bereitstellen

- Vollständigkeit / Angemessenheit?
  „[*anticipating*] *the events needed in an object's interface* [...] *is unreasonably hard*" (Sullivan, 1994)
- Abhängigkeit von Drittsoftware
  - Export der Schnittstellen für feste Middleware
- Komponenten meist nur für isolierte Nutzung entwickelt
  - Einsatz in Umgebung erfordert Modifikation

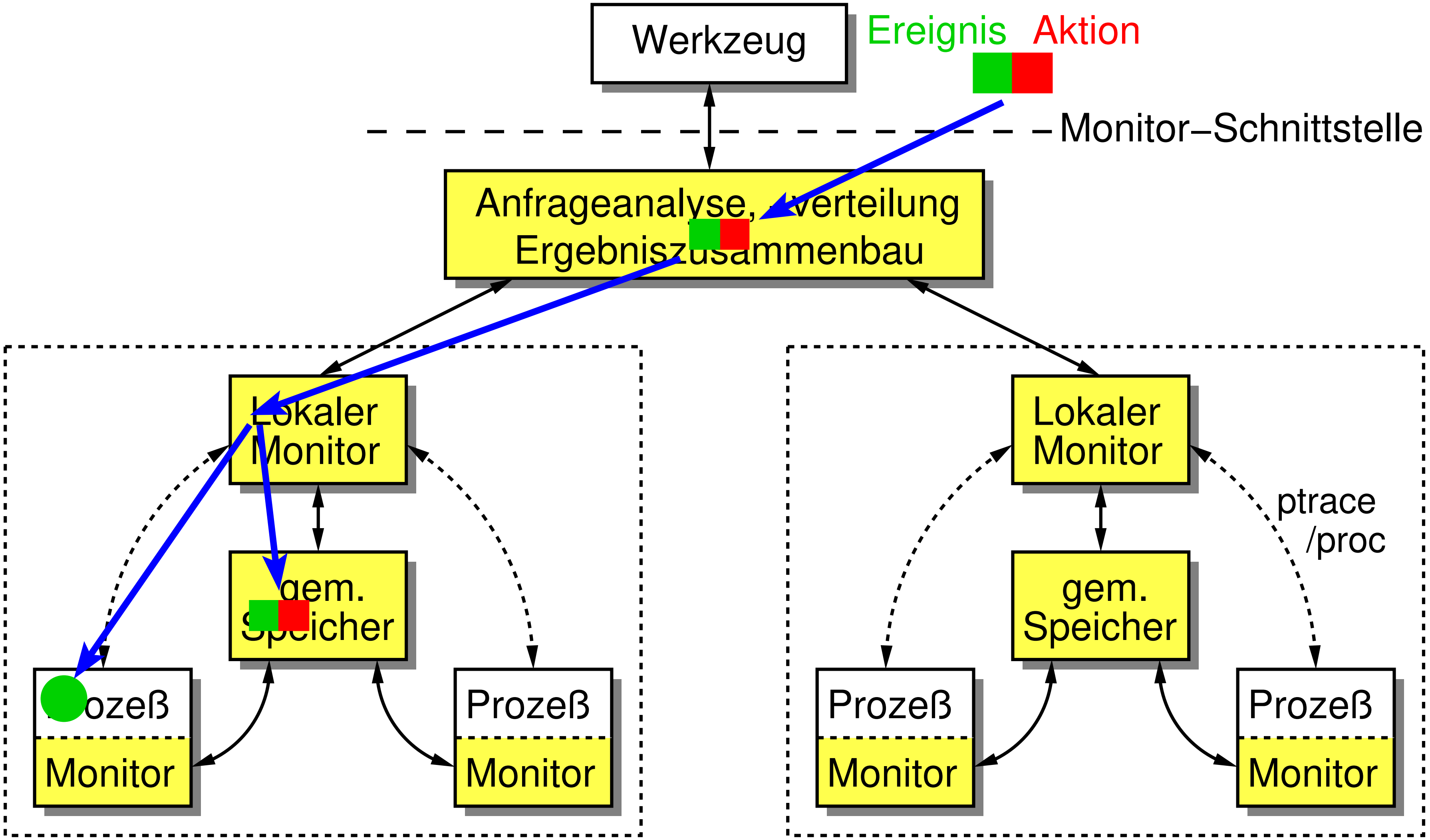
Aufbau des OCM (*OMIS Compliant Monitoring system*)



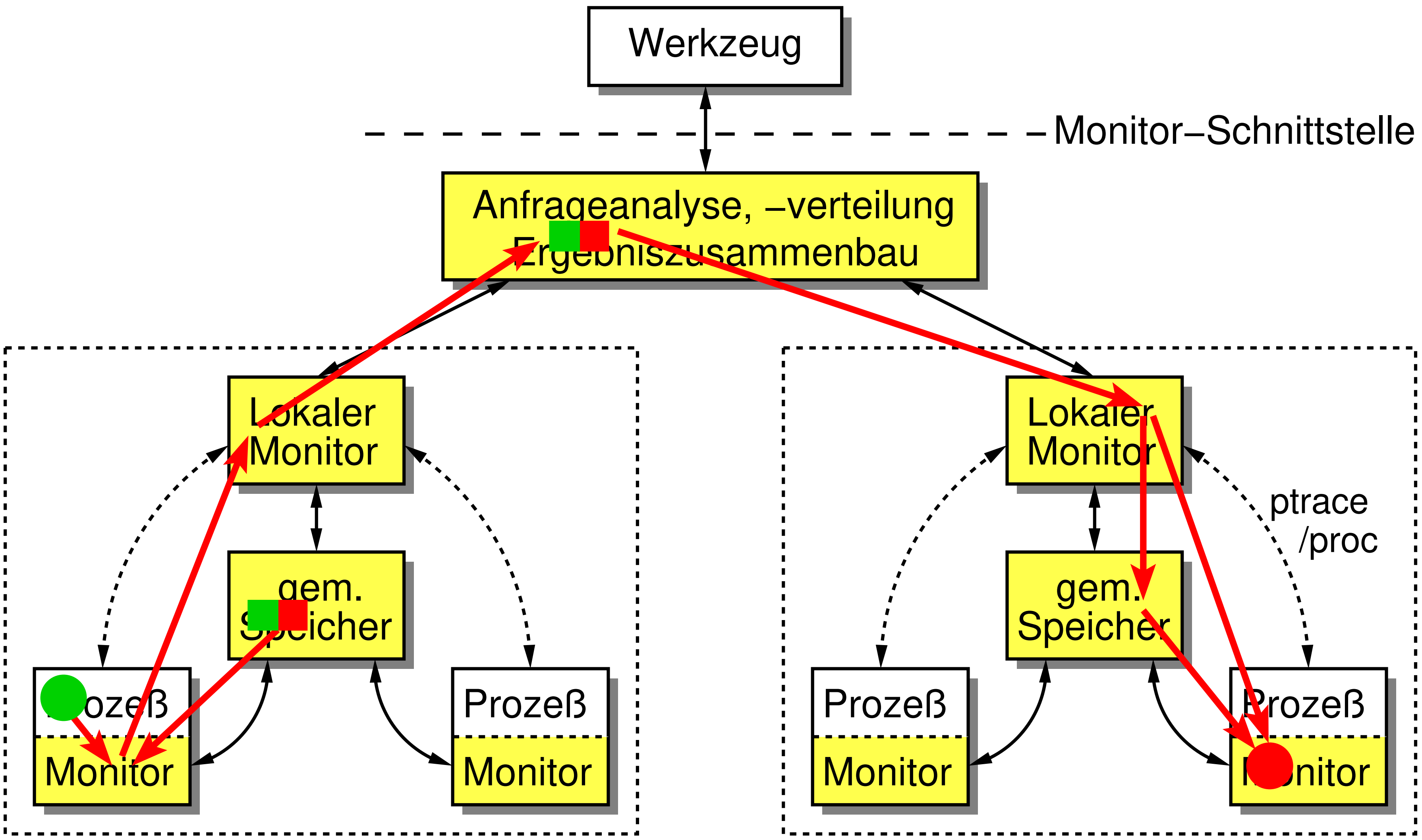
Funktion des OCM



Funktion des OCM



Funktion des OCM



Spezielle Eigenschaften des OCM:

- Überwachung von Ereig– nissen in Prozessen
- Aufruf von Aktionen in Prozessen

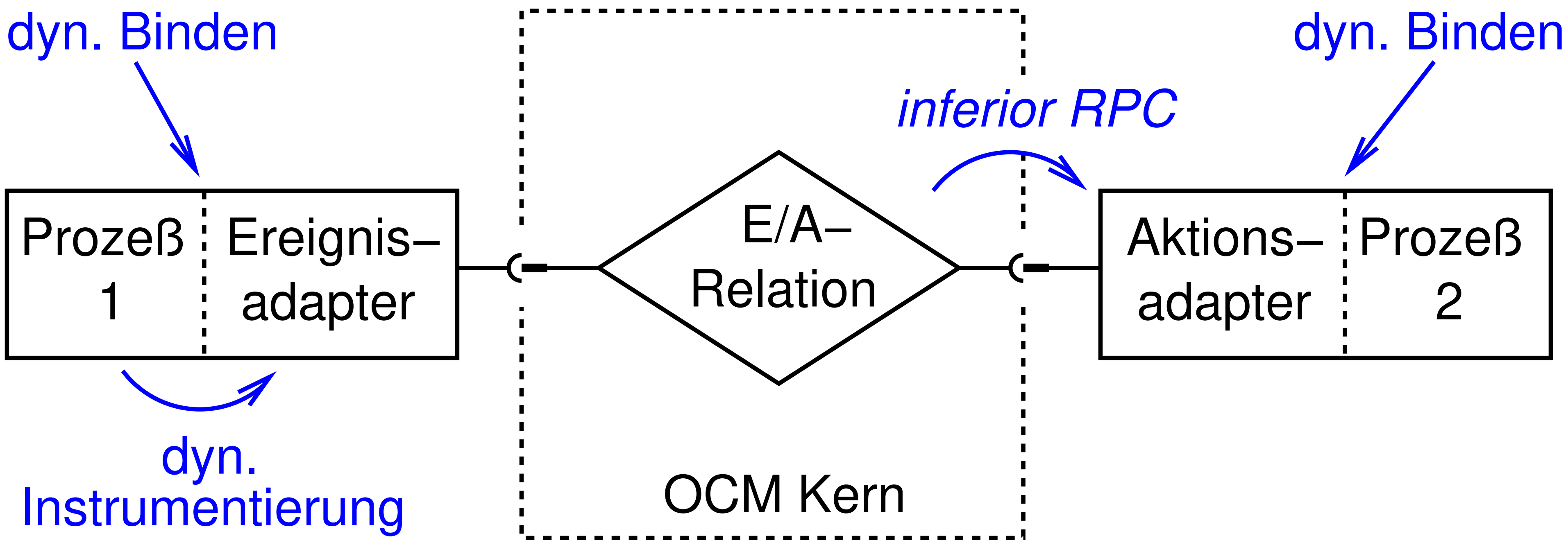
ohne spezielle Vorbereitung dieser Prozesse

- Menge der Ereignisse und Aktionen zur Laufzeit erweiterbar

Zur Realisierung:

- Dynamische Instrumentierung (*dyninst API*)
  - Kooperation mit B. P. Miller, UW Madison
- *Inferior RPC*
  - Manipulation von Keller und Befehlszähler
  - synchronisiert durch lokales Ereignis
- Dynamisches Binden
  - Aufruf des dynamischen Binders durch *inferior RPC*

Zur Realisierung:

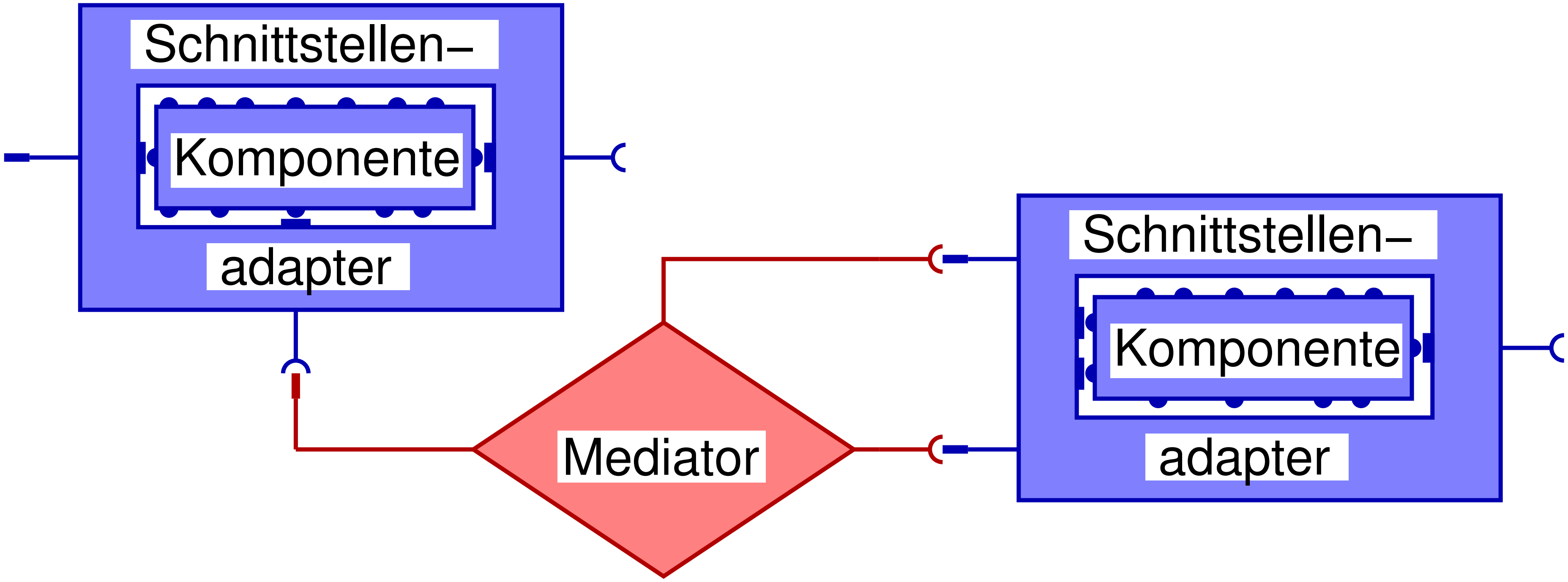


Monitorsystem $=$ dynamische Middleware

- Mediatorkonzept
  - Verhaltensregeln dynamisch programmierbar
- Dynamisches Anbinden von Schnittstellen(adaptern)
  - alle für Binder sichtbaren Symbole nutzbar
    - globale Variable / Objekte
    - globale Funktionen
    - öffentliche und geschützte Methoden
  - Ereignisse: Beginn / Ende einer Funktion bzw. Methode
  - Aktionen: nutzen Komponente wie Programmierbibliothek

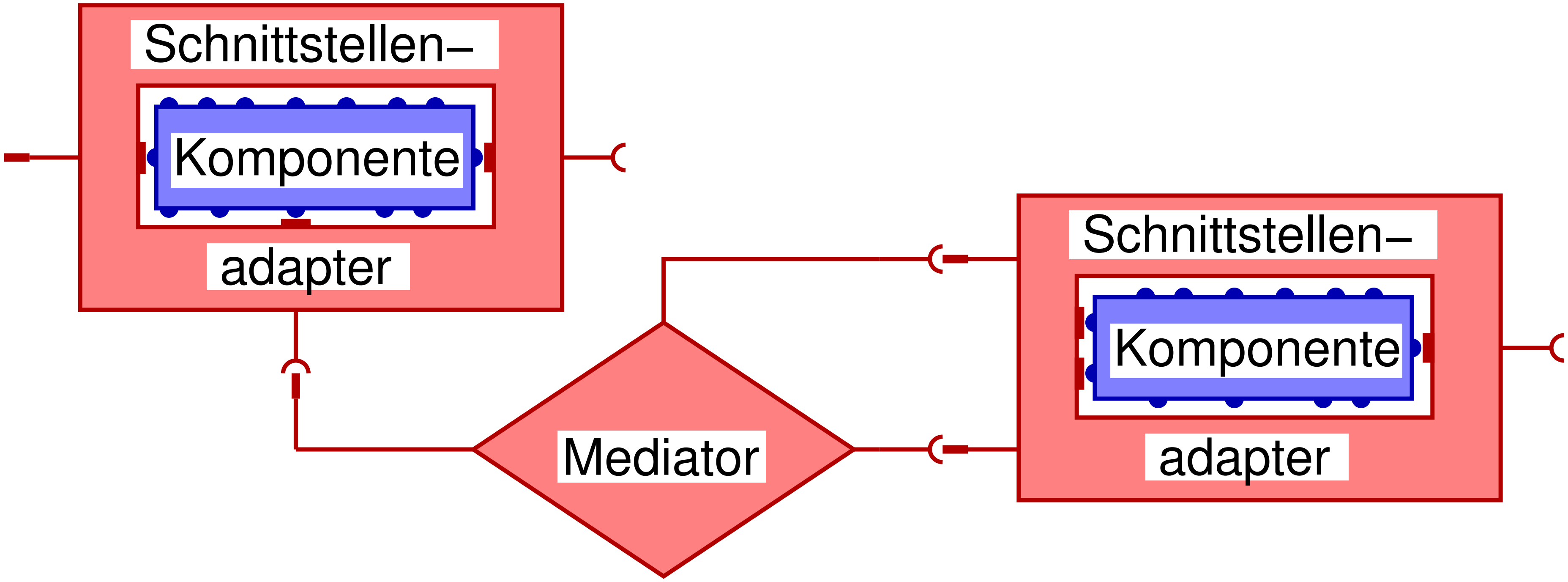
Traditionelles Vorgehen

Komponentenentwickler
Integrierer



Dynamisches Anbinden von Schnittstellen

Komponentenentwickler
Integrierer



4. Resultat

Flexiblere Nutzung von Komponenten:

- Mächtigere (interne) Schnittstelle nutzbar
- Einfache und effiziente Anpassung der Schnittstelle

Vereinfachte Entwicklung von Komponenten:

- Kein expliziter Export von Schnittstellen
  - kein unnötiger 'Ballast' im Code
  - keine Abhängigkeit von einer Middleware

Schnittstelle an sich / Dokumentation muß vorhanden sein!

Prototyp der Werkzeugumgebung

- Sichern des Programmzustands, automatisches Rücksetzen
- Konsistenz VISTOP $\leftrightarrow$ DETOP
- Zur Verhaltensintegration: 3 Ereignisse + 10 Aktionen
  - nur eine Aktion mit komplexerem Adapter
  - ohne Modifikation des Werkzeug-Quellcodes
- Aber: Datenintegration erforderte Modifikationen

☞ Bei Komponenten mit gemeinsamen Daten:
Möglichkeit einer gleichzeitigen Nuzung muß beim Entwurf berücksichtigt werden, konkrete Kooperation noch nicht

Verhaltensintegration

- Mediator-Ansatz
- Dynamisches Einbinden von Schnittstellen(adaptern)

Weitere Arbeiten

- Integration der Konzepte mit Koordinationssprache
- Unterstützung durch / Nutzung in Betriebssystemen
- Systematische Trennung von Middleware und Monitordiensten